10-1-2005

A Quasi-PTAS for Unsplittable Flow on Line Graphs

Nikhil Bansal
IBM T.J. Watson Research Center

Amit Chakrabarti
Dartmouth College

Amir Epstein
Tel Aviv University

Baruch Schieber
IBM T.J. Watson Research Center

A Quasi-PTAS for Unsplittable Flow on Line Graphs

Nikhil Bansal* Amit Chakrabarti† Amir Epstein‡ Baruch Schieber*

Dartmouth Computer Science Technical Report TR2005-561

Abstract

We study the Unsplittable Flow Problem (UFP) on a line graph, focusing on the long-standing open question of whether the problem is APX-hard. We describe a deterministic quasi-polynomial time approximation scheme for UFP on line graphs, thereby ruling out an APX-hardness result, unless NP $\subseteq$ DTIME($2^{\text{polylog}(n)}$). Our result requires a quasi-polynomial bound on all edge capacities and demands in the input instance.

Earlier results on this problem included a polynomial time $(2+\varepsilon)$-approximation under the assumption that no demand exceeds any edge capacity (the "no-bottleneck assumption") and a super-constant integrality gap if this assumption did not hold. Unlike most earlier work on UFP, our results do not require a no-bottleneck assumption.

1 Introduction

1.1 Problem Definition

The unsplittable flow problem (UFP) asks for the maximum profit subset of a given set of point-to-point flow demands that can be simultaneously routed in a given capacitated network. To be precise, the input consists of a graph $G = (V, E)$ with edge capacities $\{c_e\}_{e\in E}$ that represent the amount of a fungible resource available on each edge, and set of n 4-tuples $\{(s_i, t_i, \rho_i, w_i)\}_{i=1}^{n}$ that represent the flow demands: the i^{th} demand is to be routed from $s_i \in V$ to $t_i \in V$, requires ρ_i units of the resource and yields a profit in the amount w_i if it is routed. Each c_e, ρ_i and w_i is an integer in the range $[1, L]$, for some (large) integer L. A subset $S \subseteq \{1, \ldots, n\}$ of the demands is said to be *feasible* if all demands in S can be simultaneously routed, with each demand using one path in G (hence, "unsplittable") respecting the capacity constraints on every edge in G. The goal is to compute a feasible S that maximizes the profit $w(S) := \sum_{i\in S} w_i$.

In this paper, we study UFP on line networks. Here, the graph G is an undirected line graph with m edges. In this case we can orient the line arbitrarily to form a directed "left-to-right" path of length m and identify V with $\{0, 1, \ldots, m\}$ in the natural left-to-right order along this path. Note also that choosing a path for each routed demand is a non-issue since there is no choice in a line network (indeed, in any tree network).

*IBM T. J. Watson Research Center, Yorktown Heights, NY, USA

†Department of Computer Science, Dartmouth College, Hanover, NH 03755, USA

‡School of Computer Science, Tel Aviv University. E-Mail: amirep@tau.ac.il. Supported in part by the Israel Science Foundation.

1.2 Background and Prior Work

The unsplittable flow problem is clearly NP-complete, because when the graph G is a single edge with all demands going across it, UFP simply becomes the Knapsack problem. Two other noteworthy special cases of UFP are the Edge Disjoint Paths Problem (EDPP), obtained by setting each edge capacity and resource requirement to 1, and Uniform Capacity UFP (UCUFP), obtained by equalizing all edge capacities. EDPP has long been known to be NP-complete on directed graphs with 2 terminal pairs [FHW80] and on undirected graphs with a non-constant number of terminal pairs [Kar72]. It is also known to have an $O(\sqrt{|E|})$-approximation algorithm on directed graphs $G = (V, E)$ [Kle96]. This result was generalized to UCUFP by Srinivasan [Sri97] and eventually to UFP by Baveja and Srinivasan [BS00]. The result for UFP required the assumption $\rho_{\max} \leq c_{\min}$, which has come to be known as the *no-bottleneck assumption*. Azar and Regev provided a combinatorial and easy-to-analyze algorithm achieving the same $O(\sqrt{|E|})$ approximation ratio for UFP, under the no-bottleneck assumption. Chekuri and Khanna [CK03] refined the approximation ratio for EDPP to $O\big(\min\{|V|^{2/3}, |E|^{1/2}\}\big)$.

On the lower bound side, Guruswami et al. [GKR+03] showed that EDPP (and hence, UFP) is hard to approximate on directed graphs to a factor in $\Omega\big(|E|^{1/2-\varepsilon}\big)$ unless P = NP. Very recently, Andrews and Zhang [AZ05] gave the first nontrivial hardness result for EDPP on undirected graphs, and in follow-up work, Andrews et al. [ACKZ05] prove the current best bound of $\Omega\big(\log^{1/2-\varepsilon}|E|\big)$ under the assumption NP $\not\subseteq$ ZPTIME($2^{\text{polylog}(n)}$).

Turning to line graphs — our focus in this work — we note that UCUFP on line graphs has been studied for quite a while under various other guises, typically by viewing the line graph as a continuous *timeline* and the terminals s_i and t_i as start and finish times of a set of *jobs*. Accordingly, it has come to be known by a number of names such as "Resource Allocation," "Bandwidth Allocation," "Resource Constrained Scheduling" and "Call Admission Control." The first constant factor approximation for UCUFP on a line was given by Phillips, Uma and Wein [PUW00] who obtained a 6-approximation. This factor was then improved by Bar-Noy et al. [BBF+01] to 3 and by Calinescu et al. [CCKR01] to $(2 + \varepsilon)$. For UFP on line graphs, the first constant factor approximation was due to Chakrabarti et al. [CCGK05]; subsequent work of Chekuri, Mydlarz and Shepherd [CMS03] provided a $(2 + \varepsilon)$-approximation. Both results require the no-bottleneck assumption. The former paper proves an integrality gap of $\Omega(\log(\rho_{\max}/\rho_{\min}))$ for the natural LP relaxation when this assumption is removed.

1.3 Our Work

The main result of our work is the following theorem.

Main Theorem. *There is a quasi-polynomial time approximation scheme for* UFP *on line graphs, provided all capacities, resource requirements and profits are integers bounded by* $2^{\text{polylog}(n)}$.

Importantly, in contrast to previous work, we do not require a no-bottleneck assumption. Of course, in view of the integrality gap result of Chakrabarti et al. [CCGK05], we must achieve this by not analyzing our algorithm's solution quality by comparison to the LP optimum.

Hardness results for UFP on undirected graphs have been particularly hard to obtain. Proving APX-hardness for UFP on line graphs (indeed, even for the Resource Allocation Problem, i.e., UCUFP on a line) had been a longstanding open question. Our work resolves this open question and also hints at why hardness results for UFP in general have been hard to come by.

2 Preliminaries

We can assume, w.l.o.g., that $s_i < t_i$ for each i. By appropriate rescaling, we can assume that $\rho_{\max} = w_{\min} = 1$; this also ensures $\rho_{\min} \geq 1/L$. It is also possible to discard all demands with profit less than $(\varepsilon/n)w_{\max}$ incurring a loss of at most an ε fraction of the optimal profit; doing so ensures $w_{\max} \leq n/\varepsilon$.

For any $i \in \{1, \ldots, n\}$, the ratio w_i/ρ_i is called the *profit density* of demand i. Note that after the above adjustments we have $1 \leq w_i/\rho_i \leq Ln/\varepsilon$ for every demand i. We partition the set of demands in the input instance into classes (several of which may be empty) based on their profit densities: class q consists of demands with $2^{q-1} \leq w_i/\rho_i < 2^q$. This gives a total of at most $Q := 1 + \lfloor \lg \max_i\{w_i/\rho_i\} \rfloor \leq \text{polylog}(n)$ classes, provided $L \leq 2^{\text{polylog}(n)}$.

Any edge $\{u-1, u\} \in E$ where $u-1$ is not an s_i and u is not a t_i can be contracted, and capacities of the edges $\{u-2, u-1\}$ and $\{u, u+1\}$ adjusted, to leave the UFP instance combinatorially unchanged. After doing so repeatedly, we may assume that $m \leq n$. Thus, the complexity of an algorithm for UFP on a line network can be described in terms of n and L alone, ignoring m.

For any $i \in \{1, \ldots, n\}$, the half-open interval $(s_i, t_i]$ is called the *interval* of demand i. We say that this demand *spans* an edge $e = \{u-1, u\} \in E$ if $(s_i, t_i] \ni u$, *lies to the left* of e if $t_i \leq u-1$ and *lies to the right* of e if $s_i \geq u$. Let $S \subseteq \{1, \ldots, n\}$ be a subset of the demands. The *load* of S on edge e is defined to be the total amount of resource used by those demands in S that span e:

$$\text{load}(S, e) \;:=\; \sum_{i \in S:\, (s_i, t_i] \ni u} \rho_i \,, \text{ where } e = \{u-1, u\} \,.$$

We define a *profile* to be an m-dimensional vector indexed by the edges of G and having nonnegative real entries. For example, the edge capacities $\{c_e\}_{e \in E}$ in the given UFP instance form a profile; we denote this profile c. Another natural example is the *resource usage profile* (or simply, *profile*) of a set S of demands, defined to be the vector of its loads on all the edges of G:

$$\text{prof}(S) \;:=\; \big(\text{load}(S, \{0, 1\}),\; \text{load}(S, \{1, 2\}),\; \ldots,\; \text{load}(S, \{m-1, m\})\big) \,.$$

We use operators such as "$\leq$" and "$+$" on profiles in the standard coordinatewise manner. Therefore, we can express the feasibility of S by writing $\text{prof}(S) \leq c$. The set S is said to be a *pile* at edge e if every demand in S spans e.

It is not hard to see that the profile of a pile is a unimodal sequence. When graphed, such a profile looks like a stepped mountain. Imagine imposing a coarse uniform grid of horizontal lines on this graph and requiring the horizontal segments in the graph to lie on grid lines. Doing so greatly restricts the profile and drastically reduces its description complexity. This observation is one of the key insights behind our algorithm and it motivates the following definition.

Definition 2.1. Let $e = \{u-1, u\}$ be an edge of G and h and δ be positive reals with $h \leq c_e$, $\delta < 1$ and $1/\delta$ an integer. Let $x_1, \ldots, x_{1/\delta}$ and $y_1, \ldots, y_{1/\delta}$ be vertices of G with

$$x_1 \leq x_2 \leq \cdots \leq x_{1/\delta} \leq u-1 \quad \text{and} \quad u \leq y_{1/\delta} \leq \cdots \leq y_2 \leq y_1 \,.$$

Then the profile $(\ell_1, \ldots, \ell_m)$ where

$$\ell_i \;=\; \begin{cases} 0, & \text{for } i \leq x_1 \text{ and } i > y_1 \\ j\delta h, & \text{for } x_j < i \leq x_{j+1} \text{ and } y_{j+1} < i \leq y_j \\ h, & \text{for } x_{1/\delta} < i \leq y_{1/\delta} \end{cases}$$

is said to be a *δ-restricted profile* with *peak e* and *height h*, parametrized by the x_j's and y_j's. This particular profile is denoted $\mathrm{RP}_\delta(e; h; x_1, \ldots, x_{1/\delta}; y_1, \ldots, y_{1/\delta})$.

Note that by not requiring the x_j's and y_j's to be distinct, we allow the "step size" of such a restricted profile at one of these vertices to be greater than δh: it can be a larger multiple of δh.

It is clear from the definition that the number of δ-restricted profiles in an m-edge line graph with a given peak and a given height is upper bounded by $m^{2/\delta}$. This fact will be useful in the sequel.

3 Two Lemmas About Restricted Profiles

Consider any optimal solution $S^* \subseteq \{1, \ldots, n\}$ to the given UFP instance and any edge $e \in E$. The subset of demands in S^* that span e form a pile; let T^* be such a subset. In this section we prove two lemmas that together show how to compute, in polynomial time, a set T of demands that yields a $(1 - O(\delta))$ fraction of the profit of T^* and whose profile is δ-restricted and approximates the profile of T^*, for some appropriate small δ. The lemmas need some additional restrictions on the demands in T^* that are made precise below.

Throughout this section we shall assume that $0 < \delta < 1$ and that $1/\delta$ is an integer.

Lemma 3.1. *Let S be a pile at edge e of class-q demands such that $\rho_i \leq B$ for all $i \in S$. Let h be the largest integer multiple of $\rho_{\min}$ that does not exceed* $\mathrm{load}(S, e)$. *Then there exists a δ-restricted profile π with peak e and height h and a subset $T \subseteq S$ such that*

1. $\mathrm{prof}(T) \leq \pi \leq \mathrm{prof}(S)$, *and*
2. $w(S \setminus T) \leq 2^{q+1}(\delta h + B)$.

Proof. The idea is to "scan" the edges of G from left to right and mark off the first edge on which the load of S is at least δh, at least $2\delta h$, etc.; these edges then define the left half of the profile π. To be precise, define

$$\begin{aligned} x_j &= \min\{i : \mathrm{load}(S, \{i, i+1\}) \geq j\delta h\}, \quad \text{for } 1 \leq j \leq 1/\delta, \\ y_j &= \max\{i : \mathrm{load}(S, \{i-1, i\}) \geq j\delta h\}, \quad \text{for } 1 \leq j \leq 1/\delta. \end{aligned}$$

and let $\pi := \mathrm{RP}_\delta(e; h; x_1, \ldots, x_{1/\delta}; y_1, \ldots, y_{1/\delta})$. It follows from the construction that $\pi \leq \mathrm{prof}(S)$.

It also follows that every entry of the vector $(\mathrm{prof}(S) - \pi)$ is upper bounded by δh, except for the entries indexed by edges between vertices $x_{1/\delta}$ and $y_{1/\delta}$, which are upper bounded by $\rho_{\min}$. Thus, to construct a subset $T \subseteq S$ such that $\mathrm{prof}(T) \leq \pi$, we can apply the following greedy procedure. Order all demands in S by their left end-points (i.e., s_i) and greedily select demands until the total resource requirement of the selected demands is at least δh; let A be the set of demands thus selected. Then order all demands by their right end-points (i.e., t_i) and do the same; let B be the resulting set. Let $T := S \setminus (A \cup B)$.

We now have a set T that satisfies property 1. But note that the total resource requirement of the demands in A is at most $\delta h + B$, because each individual demand $i \in S$ has $\rho_i \leq B$. The same is true for B. Since all demands under consideration are in class q, the total profit in $A \cup B$ is at most $2(\delta h + B) \cdot 2^q = 2^{q+1}(\delta h + B)$, which shows that T satisfies property 2. □

We now present a linear programming based algorithm to pack a high-profit subset of a given set of demands into a given restricted profile and prove a basic property of this algorithm.

Algorithm 1: PILE-PACK

Input: line graph $G = (V, E)$, profile $\pi = \mathrm{RP}_\delta(e; h; x_1, \ldots, x_{1/\delta}; y_1, \ldots, y_{1/\delta})$, set $S \subseteq [n]$ indexing demands $\{(s_i, t_i, w_i, \rho_i) : 1 \le i \le n\}$

Output: a subset $T \subseteq S$ with $\mathrm{prof}(T) \le \pi$

1 Delete from S all demands i where $s_i < x_1$ or $t_i > y_1$

2 **for** $j = 1$ *to* $1/\delta$ **do**

3 $\quad A_j \leftarrow \{i \in S : s_i \ge x_j\}$

4 $\quad B_j \leftarrow \{i \in S : t_i \le y_j\}$

5 Compute a vertex solution to the following linear program in the variables $\{\alpha_i\}_{i \in S}$:

$$\begin{aligned} &\text{maximize } \textstyle\sum_{i \in S} w_i \alpha_i\,, && \text{for nonnegative } \alpha_i, \text{ s.t.} \\ &\textstyle\sum_{i \in A_j} \rho_i \alpha_i \;\le\; j\delta h\,, && \text{for } 1 \le j \le 1/\delta && (3.1) \\ &\textstyle\sum_{i \in B_j} \rho_i \alpha_i \;\le\; j\delta h\,, && \text{for } 1 \le j \le 1/\delta && (3.2) \\ &\alpha_i \;\le\; 1\,, && \text{for } i \in S && (3.3) \end{aligned}$$

6 $T \leftarrow \{i \in S : \alpha_i = 1\}$

7 **return** T

Lemma 3.2. *Let S be a pile at edge e of class-q demands such that $\rho_i \le B$ for all $i \in S$, and let π be a δ-restricted profile with peak e. Let T^* be a maximum profit subset of S such that* $\mathrm{prof}(T^*) \le \pi$ *and let T be the set returned by* PILE-PACK(G, π, S). *Then $w(T^*) - w(T) \le 2^{q+1} B/\delta$.*

Proof. Assume, w.l.o.g., that no deletions are necessary in Step 1 of PILE-PACK. Note that if the linear program (3.1)–(3.3) were solved with the added condition $\alpha_i \in \{0, 1\}$ for $i \in S$, then the set returned by PILE-PACK would be a maximum profit subset of S that fit the profile π. Therefore, if W denotes the (fractional) optimum of the LP, we have $w(T^*) \le W$.

A vertex solution of the LP must make $|S|$ inequalities tight. The total number of distinct inequalities given by (3.1) and (3.2) is at most $2/\delta$. Therefore, at most $2/\delta$ of the $|S|$ variables $\{\alpha_i\}$ can be fractional and cause the corresponding demands to be discarded from the fractional solution when forming the set T. Each of these discarded demands has resource requirement at most B, and thus (since it is a class-q demand) profit at most $2^q B$. Therefore $W - w(T) \le (2B/\delta) \cdot 2^q = 2^{q+1} B/\delta$ and the lemma follows. □

4 The Final Algorithm

We are ready to describe our algorithm for UFP on a line graph. We begin with some notation and give an intuitive outline of the algorithm.

For each edge $e = \{u - 1, u\}$ of the graph G, let $G_{L,e}$ denote the subgraph of G induced by the vertices $\{0, \ldots, u-1\}$ and let $G_{R,e}$ denote the subgraph of G induced by the vertices $\{u, u+1, \ldots, m\}$. Let LEFT(e), RIGHT(e) and SPAN(e) denote the subset of demands in the input that (respectively) lie to the left of e, lie to

the right of e and span e. Recall that the demands in the input are partitioned into $Q \leq \text{polylog}(n)$ classes based on their profit densities. Consider the set of class-q demands that span e in an optimal solution. Some of these demands are "large" (have a high resource requirement) whereas the rest are "small." The large demands cannot be too many in number, so the algorithm can try out all possible subsets till it hits the "right" one. As for the small demands, they form a pile at e and, by Lemmas 3.1 and 3.2, the profile of such a pile can be approximated by an appropriate δ-restricted profile with peak e, with the smallness of the demands ensuring that the profit reduction from such an approximation is tiny. Our algorithm tries out all possible settings of the parameters of such a profile; as seen before, there are not too many, and this upper bounds the running time.

Algorithm 2: LINE-UFP-RECURSIVE

Input: line graph $G = (V, E)$, edge capacities $\{c_e : e \in E\}$, set $S \subseteq [n]$ indexing demands $\{(s_i, t_i, w_i, \rho_i) : 1 \leq i \leq n\}$, real parameter $\delta > 0$

Output: a routable subset of S with profit $\geq (1 - O(\delta))\text{OPT}$

1 **if** $|S| \leq 1$ **then**
2 **if** $\text{prof}(S) \leq c$ **then return** S **else return** $\emptyset$
3 find an edge $e^* \in E$ such that $|\text{LEFT}(e^*)| \leq n/2$ and $|\text{RIGHT}(e^*)| \leq n/2$
4 **for** $q = 1$ *to* Q **do** $S_q \leftarrow \{i \in \text{SPAN}(e^*) : 2^{q-1} \leq w_i/\rho_i \leq 2^q\}$
5 **foreach** *Q-tuple* $(T_1, \ldots, T_Q)$ *with each* $T_q \subseteq S_q$ *and* $|T_q| \leq 1/\delta^2$ **do**
6 $T \leftarrow \bigcup_{q=1}^{Q} T_q$
7 **if** *all demands in* T *can be routed* **then**
8 route all demands in T and obtain residual capacities $\{c'_e : e \in E\}$
9 **foreach** $(h'_1, \ldots, h'_Q) \in \mathbb{R}^q$ *with each* h'_q *an integer multiple of* $\rho_{\min}$ *and* $h'_1 + \cdots + h'_Q \leq c'_{e^*}$ **do**
10 **for** $q = 1$ *to* Q **do** $S_{q,\text{small}} \leftarrow \{i \in S_q \setminus T_q : \rho_i \leq \delta^2(h'_q + \rho_{\min} + \text{load}(T_q, e^*))\}$
11 **foreach** *Q-tuple* $(\pi_1, \ldots, \pi_Q)$ *with each* π_q *a* δ*-restricted profile with peak* e^* *and height* h'_q*, such that* $\pi_1 + \cdots + \pi_Q \leq c'$ **do**
12 **for** $q = 1$ *to* Q **do** $U_q \leftarrow \text{PILE-PACK}(G, \pi_q, S_{q,\text{small}})$
13 $U \leftarrow \bigcup_{q=1}^{Q} U_q$
14 route all demands in U_q and obtain residual capacities $\{c''_e : e \in E\}$
15 $L \leftarrow \text{LINE-UFP-RECURSIVE}(G_{L,e^*}, c'', \text{LEFT}(e^*))$
16 $R \leftarrow \text{LINE-UFP-RECURSIVE}(G_{R,e^*}, c'', \text{RIGHT}(e^*))$
17 record the solution $T \cup U \cup L \cup R$
18 **return** the most profitable of the recorded solutions

The following two lemmas establish the key properties of this algorithm.

Lemma 4.1. *Algorithm* LINE-UFP-RECURSIVE *runs in time quasi-polynomial in n, provided L is quasi-polynomial in n.*

Proof. We upper bound the number of iterations of each of the three most complex nested loops. There are at most n^{1/δ^2} possibilities for each set T_q, so the loop beginning at line 5 runs for at most n^{Q/δ^2} iterations. Since $c_{\max}/\rho_{\min} \leq L$, the loop beginning at line 9 runs for at most L^Q iterations. Finally, there are at most

$n^{2/\delta}$ possibilities for each δ-restricted profile π_q. Thus, the loop beginning at line 11 runs for at most $n^{2Q/\delta}$ iterations.

Putting it all together, the algorithm makes at most $2n^{Q/\delta^2+2Q/\delta}L^Q$ recursive calls to itself, with subproblems of size at most $n/2$ each and uses an additional poly(n) processing time. Recalling that $Q = 1 + \lfloor \lg \max_i\{w_i/\rho_i\} \rfloor \leq O(\log(Ln/\varepsilon))$ and using $L \leq 2^{\text{polylog}(n)}$, we see that the overall running time is bounded by $2^{\text{polylog}(n)}$, as claimed. □

Lemma 4.2. *Let $\mathcal{O}$ be an optimal solution to the given* UFP *instance and δ be a small enough positive real such that $1/\delta$ is an integer. Algorithm* LINE-UFP-RECURSIVE *returns a feasible solution with profit at least $(1 - 13\delta)w(\mathcal{O})$.*

Proof. We proceed by induction on $|S|$. The algorithm clearly returns an optimal solution if $|S| \leq 1$, so we focus on the case $|S| > 1$.

Consider the edge e^* identified by the algorithm in line 3 and the sets S_q of class-q demands that span e^*. Let $A_q := S_q \cap \mathcal{O}$ be the subset of S_q routed by $\mathcal{O}$ and let $h_q := \text{load}(A_q, e^*)$. Also, define the sets

$$\begin{aligned} A_{q,\text{large}} &:= \{i \in A_q : \rho_i > \delta^2 h_q\}, \\ A_{q,\text{small}} &:= \{i \in A_q : \rho_i \leq \delta^2 h_q\}. \end{aligned}$$

Since the demands in $A_{q,\text{large}}$ all span e^* and have total load at most h_q on e^*, there can be at most $1/\delta^2$ of them; so the algorithm will eventually set $T_q = A_{q,\text{large}}$, for each q. From now on, we focus on only this iteration of the loop beginning at line 5.

Consider any arbitrary q and apply Lemma 3.1 to the set $A_{q,\text{small}}$, noting that $B = \delta^2 h_q$ is a bound on the ρ_i's. The algorithm will eventually pick h'_q to be the largest integer multiple of $\rho_{\min}$ not exceeding $\text{load}(A_{q,\text{small}}, e^*)$ and π_q to be the δ-restricted profile whose existence is guaranteed by Lemma 3.1. Let us concentrate on these choices of h'_q and π_q. We have

$$h'_q + \text{load}(T_q, e^*) \;\leq\; h_q \;<\; h'_q + \rho_{\min} + \text{load}(T_q, e^*). \tag{4.4}$$

Let W be the maximum possible profit of a subset of $A_{q,\text{small}}$ that fits profile π_q. Then

$$W \;\geq\; w(A_{q,\text{small}}) - 2^{q+1}(\delta h'_q + B) \;\geq\; w(A_{q,\text{small}}) - 2^{q+1}(\delta + \delta^2)h_q,$$

where we used the left inequality in (4.4). Now consider the construction of the set $S_{q,\text{small}}$ in line 10; the right inequality in (4.4) ensures that it is a superset of $A_{q,\text{small}}$. Moreover, for any $i \in S_{q,\text{small}}$, we have $\rho_i \leq \delta^2(h_q + \rho_{\min}) \leq 2\delta^2 h_q$. Lemma 3.2 says that when the algorithm calls PILE-PACK with this profile π_q and the set of demands in $S_{q,\text{small}}$, it will obtain a set U_q with

$$w(U_q) \;\geq\; W - 2^{q+1}(2\delta^2 h_q)/\delta \;\geq\; w(A_{q,\text{small}}) - 2^{q+1}(3\delta + \delta^2)h_q \;\geq\; w(A_{q,\text{small}}) - 13\delta \cdot 2^{q-1}h_q,$$

where the final inequality uses the assumption that δ is small enough. Thus, we have $w(T_q) + w(U_q) \geq w(A_q) - 13\delta \cdot 2^{q-1}h_q \geq (1 - 13\delta)w(A_q)$, since each demand in S_q has profit density at least 2^{q-1}. Since q was arbitrary, the algorithm will eventually satisfy these conditions for all q.

Lemmas 3.1 and 3.2 also guarantee that for each q, $\text{prof}(U_q) \leq \pi_q \leq \text{prof}(A_{q,\text{small}})$. Therefore, routing $T \cup U$ leaves at least as much residual capacity on each edge as does routing $\bigcup_{q=1}^{Q} A_q$. It follows that the two recursive calls work on graphs with "sufficient" residual capacity. Let $\mathcal{O}_L$ and $\mathcal{O}_R$ denote the subsets of $\mathcal{O}$

consisting of demands that lie to the left of e and to the right of e, respectively. By induction hypothesis, the recursive calls return sets L and R with $w(L) \geq (1-13\delta)w(\mathcal{O}_L)$ and $w(R) \geq (1-13\delta)w(\mathcal{O}_R)$. Therefore, Algorithm LINE-UFP-RECURSIVE will eventually record a solution with profit at least $(1-13\delta)w(\mathcal{O})$. □

Theorem 4.3 (Main Theorem restated)**.** *There is a quasi-polynomial time approximation scheme for* UFP *on line graphs, provided all capacities, resource requirements and profits are integers bounded by* $2^{\text{polylog}(n)}$.

Proof. This follows immediately from Lemmas 4.1 and 4.2. □

5 Concluding Remarks

We have provided a quasi-PTAS for UFP on line graphs, thereby virtually ruling out an APX-hardness result for the problem. Unlike most earlier work, we do not require a no-bottleneck assumption.

An immediate open question is whether our result extends to more general classes of graphs, such as trees. Determining the simplest class of graphs for which UFP is APX-hard remains a most interesting open problem.

References

[ACKZ05] Matthew Andrews, Julia Chuzhoy, Sanjeev Khanna, and Lisa Zhang. Hardness of the undirected edge-disjoint paths problem with congestion. In *Proc. 46th Annual IEEE Symposium on Foundations of Computer Science*, 2005. to appear.

[AZ05] Matthew Andrews and Lisa Zhang. Hardness of the undirected edge-disjoint paths problem. In *Proc. 37th Annual ACM Symposium on the Theory of Computing*, pages 276–283, 2005.

[BBF+01] Amotz Bar-Noy, Reuven Bar-Yehuda, Ari Freund, Joseph (Seffi) Naor, and Baruch Schieber. A unified approach to approximating resource allocation and scheduling. *J. ACM*, 48(5):1069–1090, 2001. Preliminary version in *Proc. 32nd Annu. ACM Symp. Theory Comput.*, pages 735–744, 2000.

[BS00] Alok Baveja and Aravind Srinivasan. Approximation algorithms for disjoint paths and related routing and packing problems. *Math. Oper. Res.*, 25(2):255–280, 2000.

[CCGK05] Amit Chakrabarti, Chandra Chekuri, Anupam Gupta, and Amit Kumar. Approximation algorithms for the unsplittable flow problem. *Algorithmica*, 2005. To appear; preliminary version in *Proc. 5th Intl. Workshop on Approximation Algorithms for Combinatorial Optimization*, pages 51–66, 2002.

[CCKR01] Gruia Calinescu, Amit Chakrabarti, Howard Karloff, and Yuval Rabani. Improved approximation algorithms for resource allocation. In *Proceedings of the 9th Integer Programming and Combinatorial Optimization Conference*, volume 2337 of *Lecture Notes in Computer Science*, pages 439–456, 2001.

[CK03] Chandra Chekuri and Sanjeev Khanna. Edge disjoint paths revisited. In *Proc. 14th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 628–637, 2003.

[CMS03] Chandra Chekuri, Marcelo Mydlarz, and F. Bruce Shepherd. Multicommodity demand flow in a tree. In *Proc. 30th International Colloquium on Automata, Languages and Programming*, pages 410–425, 2003.

[FHW80] Steve Fortune, John E. Hopcroft, and James Wylie. The directed subgraph homeomorphism problem. *Theor. Comput. Sci.*, 10:111–121, 1980.

[GKR+03] Venkatesan Guruswami, Sanjeev Khanna, Rajmohan Rajaraman, F. Bruce Shepherd, and Mihalis Yannakakis. Near-optimal hardness results and approximation algorithms for edge-disjoint paths and related problems. *J. Comput. Syst. Sci.*, 67(3):473–496, 2003. Preliminary version in *Proc. 31st Annu. ACM Symp. Theory Comput.*, pages 19–28, 1999.

[Kar72] Richard M. Karp. Reducibility among combinatorial problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, New York, NY, 1972.

[Kle96] Jon M. Kleinberg. *Approximation Algorithms for Disjoint Paths Problems*. PhD thesis, MIT, 1996.

[PUW00] Cynthia Phillips, R. N. Uma, and Joel Wein. Off-line admission control for general scheduling problems. In *Proc. 11th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 879–888, 2000.

[Sri97] Aravind Srinivasan. Improved approximations for edge-disjoint paths, unsplittable flow, and related routing problems. In *Proceedings of the 38th Annual IEEE Symposium on Foundations of Computer Science*, pages 416–425, 1997.